

🧭 Uso de OSRM (Open Source Routing Machine) para cálculo de distancias en el TPI

🎯 Objetivo

Este documento tiene como objetivo guiar paso a paso la instalación, configuración y uso del sistema **OSRM (Open Source Routing Machine)** para calcular distancias entre ubicaciones geográficas dentro de Argentina, en el contexto del *Trabajo Práctico Integrador* de la asignatura *Backend de Aplicaciones*.

El servicio se instalará como **contenedor Docker** dentro del `docker-compose` del proyecto y expondrá un endpoint HTTP local para calcular rutas.

1. 🌎 ¿Qué es OSRM?

OSRM (Open Source Routing Machine) es una herramienta open-source de cálculo de rutas geográficas basada en los datos libres de [OpenStreetMap](#). Permite:

- Calcular rutas reales por calle entre ubicaciones
- Obtener distancias y tiempos estimados de viaje
- Ejecutar localmente en un contenedor sin depender de servicios de terceros

En nuestro caso, se utilizará para calcular distancias entre depósitos, orígenes y destinos de contenedores en rutas terrestres de Argentina, de forma gratuita y offline desde un microservicio Spring Boot.

2. 🗺 Recolección de datos geográficos (Argentina)

Mínimos **Requisitos** para la ejecución:

- Tener Docker instalado
- Conexión a Internet para la descarga inicial de mapas
- 3 GB de espacio libre para el mapa completo de Argentina

1. Crear un directorio para los datos:

```
mkdir -p osrm-data && cd osrm-data
```

2. Descargar el archivo `.osm.pbf` de Argentina desde [Geofabrik](#):

```
wget https://download.geofabrik.de/south-america/argentina-latest.osm.pbf
```

[!TIP] Validar que se descarguen los 383 MB del archivo.

```

philip: ~ -> llack > pop-osrm > osrm-data
$ wget -c https://download.geofabrik.de/south-america/argentina-latest.osm.pbf
--2025-10-29 21:49:47-- https://download.geofabrik.de/south-america/argentina-latest.osm.pbf
Resolving download.geofabrik.de (download.geofabrik.de)... 95.216.245.233, 95.217.45.61, 95.217.63.98, ...
Connecting to download.geofabrik.de (download.geofabrik.de) |95.216.245.233|:443... connected.
HTTP request sent, awaiting response... 200 OK
Location: https://download.geofabrik.de/south-america/argentina-251029.osm.pbf [following]
--2025-10-29 21:49:48-- https://download.geofabrik.de/south-america/argentina-251029.osm.pbf
Reusing existing connection to download.geofabrik.de:443.
HTTP request sent, awaiting response... 200 OK
Length: 401487500 (383M) [application/octet-stream]
Saving to: 'argentina-latest.osm.pbf'

argentina-latest.osm.pbf                                              100%[=====] 382.89M  8.70MB/s    in 65s

2025-10-29 21:50:53 (5.92 MB/s) - 'argentina-latest.osm.pbf' saved [401487500/401487500]

```

3. ⚙️ Procesamiento de datos con perfil CH (Contraction Hierarchies)

Si usás una Mac con chip Apple (M1/M2/M3), asegurate de correr las imágenes Docker con `--platform=linux/arm64`

Para preparar los datos, hay que realizar dos pasos:

Paso 1: Extraer datos

osrm-extract para Windows / Linux con amd86

```
docker run -t -v $(pwd)/osrm-data:/data osrm/osrm-backend osrm-extract -p /opt/car.lua /data/argentina-latest.osm.pbf
```

osrm-extract para MacOS con Apple Silicon

```
docker run --rm \
--platform=linux/arm64 \
-v $(pwd):/data \
ghcr.io/project-osrm/osrm-backend:latest \
osrm-extract -p /opt/car.lua /data/argentina-latest.osm.pbf
```

Paso 2: Generar jerarquía de ruteo (contract)

osrm-contract para Windows / Linux con amd86

```
docker run -t -v $(pwd)/osrm-data:/data osrm/osrm-backend osrm-contract /data/argentina-latest.osm.pbf
```

osrm-contract para MacOS con Apple Silicon

```
docker run --rm \
--platform=linux/arm64 \
-v $(pwd):/data \
```

```
ghcr.io/project-osrm/osrm-backend:latest \
osrm-contract /data/argentina-latest.osrm
```

[!TIP] En una máquina promedio este proceso demora unos 15 minutos.

🧠 Este paso genera archivos adicionales que permiten responder rutas de forma muy eficiente

4. 🚀 Levantar servicio de rutas con Docker Compose

Crear un archivo `docker-compose.osrm.yml` en el raíz del proyecto:

```
services:
  osrm:
    image: ghcr.io/project-osrm/osrm-backend:latest
    platform: linux/arm64
    container_name: osrm
    ports:
      - "5000:5000"
    volumes:
      - ./osrm-data:/data
    command: >
      osrm-routed /data/argentina-latest.osrm
```

Levantar el contenedor:

```
docker compose -f docker-compose.osrm.yml up
```

💡 Si el puerto 5000 está ocupado, podés cambiarlo a otro (por ejemplo `5001:5000`) y acceder en <http://localhost:5001>

5. 🛰 Consultar distancia entre dos coordenadas

Con el servicio corriendo, se puede hacer una consulta HTTP:

```
curl
"http://localhost:5000/route/v1/driving/-64.18105,-31.4135;-60.6985,-32.94
71?overview=false"
```

⌚ Los parámetros son: `long, lat` separados por ; entre origen y destino.

Respuesta esperada:

```
{
  "routes": [
```

```
{  
    "distance": 351000.3,  
    "duration": 15300.7,  
    ...  
}  
,  
"code": "Ok"  
}
```

La distancia está en **metros** y el tiempo en **segundos**.

🔄 Alternativa: uso del modo MLD (Multi-Level Dijkstra)

Si en algún momento quisieras cambiar a un modo de enrutamiento más flexible:

```
# Extraer  
osrm-extract -p /opt/car.lua ...  
# Preparar  
osrm-partition ...  
osrm-customize ...  
# Ejecutar  
osrm-routed --algorithm mld ...
```

Pero **para el TPI** recomendamos mantener el modo por defecto (CH) por su simplicidad y eficiencia.